

# Kernel functions based on triplet similarity comparisons

Matthäus Kleindessner  
Department of Computer Science  
University of Tübingen  
72076 Tübingen, Germany

Ulrike von Luxburg  
Department of Computer Science  
University of Tübingen  
72076 Tübingen, Germany

## Abstract

We propose two ways of defining a kernel function on a data set when the only available information about the data set are similarity triplets of the form “Object A is more similar to object B than to object C”. Studying machine learning and data mining problems based on such restricted information has become very popular in recent years since it can easily be provided by humans via crowd sourcing. While previous approaches try to construct a low-dimensional Euclidean embedding of the data set that reflects the given similarity triplets, we aim at defining meaningful kernel functions on the data set that correspond to high-dimensional embeddings. These kernel functions can subsequently be used to apply all the standard kernel methods to solve tasks such as clustering, classification or principal component analysis on the data set.

**Keywords:** ordinal data, comparison-based algorithms, similarity triplets, ordinal embedding, non-metric multidimensional scaling, kernel functions

## Introduction

Many machine learning algorithms are based on a notion of similarity between objects. The rationale is that objects that are similar to each other tend to have the same class label, belong to the same clusters, and so on. However, in complex tasks, like estimating suitable age ratings for movies, it can be hard to come up with a meaningful similarity function that can be evaluated automatically. On the other hand, in such scenarios humans often intuitively know which objects should be considered similar, and it is then natural to incorporate human expertise into the machine learning process. One way of doing so is to collect statements like “Movie *A* is more similar to movie *B* than to movie *C*” from people. We refer to such statements as similarity triplets and formally define them as binary answers to distance comparisons

$$d(A, B) \stackrel{?}{<} d(A, C), \quad (1)$$

where  $d$  is a symmetric dissimilarity function on some set  $\mathcal{X}$  and  $A, B, C \in \mathcal{X}$ . To simplify presentation, we may assume that either  $d(A, B) < d(A, C)$  or  $d(A, B) > d(A, C)$  holds true whenever  $B \neq C$ . We refer to object  $A$  as the anchor object in the distance comparison (1). Note that a similarity triplet might be incorrect, meaning that it claims a positive answer to the comparison (1) although in fact it holds that  $d(A, B) > d(A, C)$ . It is widely accepted that humans are better and more reliable in providing similarity triplets, which means assessing similarity on a relative scale, than in providing numerical similarity estimates on an absolute scale (“The similarity between movie  $A$  and movie  $B$  is 0.8”). Due to the emergence of crowd sourcing platforms, collecting similarity triplets has been tremendously facilitated in recent years.

Given a data set  $\mathcal{D}$  and similarity triplets for its objects, it is not clear how to solve machine learning problems on  $\mathcal{D}$ . A general approach is to construct an ordinal embedding of  $\mathcal{D}$ , that is to map objects to a Euclidean space of a small dimension such that the given similarity triplets are preserved as well as possible (Agarwal et al., 2007; Tamuz et al., 2011; van der Maaten and Weinberger, 2012; Kleindessner and von Luxburg, 2014; Terada and von Luxburg, 2014; Arias-Castro, 2015; Amid and Ukkonen, 2015). Once such an ordinal embedding is constructed, one can simply solve a problem on  $\mathcal{D}$  by solving it on the ordinal embedding. Only recently, algorithms have been proposed for solving a number of specific problems directly without constructing an ordinal embedding in an intermediate step (Heikinheimo and Ukkonen, 2013; Ukkonen et al., 2015; Kleindessner and von Luxburg, 2016). With this paper we provide another generic means for solving machine learning problems based on similarity triplets that is different from the ordinal embedding approach. We define two data-dependent kernel functions on  $\mathcal{D}$ , corresponding to high-dimensional embeddings of  $\mathcal{D}$ , that can subsequently be used by any kernel method. Our proposed kernel functions measure similarity between two objects in  $\mathcal{D}$  by comparing to which extent the two objects give rise to similar similarity triplets. The hope is that in doing so we quantify the relative difference in the locations of the two objects in  $\mathcal{D}$ . Our experiments on both artificial and real data show that this is indeed the case and that the similarity scores defined by our kernel functions are meaningful.

## Our kernel functions

Let  $\mathcal{X}$  be an arbitrary set and  $d : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}_0^+$  be a symmetric dissimilarity function on  $\mathcal{X}$ : a higher value of  $d$  means that two elements of  $\mathcal{X}$  are more dissimilar (or, likewise, less similar) to each other. Assume we are given a collection  $\mathcal{S}$  of similarity triplets about objects of some data set  $\mathcal{D} = \{x_1, \dots, x_n\} \subseteq \mathcal{X}$ , which are encoded as follows: an ordered triple of distinct objects  $(x_i, x_j, x_k) \in \mathcal{S}$  is interpreted as  $d(x_i, x_j) < d(x_i, x_k)$ . Similarity triplets in  $\mathcal{S}$  can be incorrect (compare with the previous section), but for the moment assume that contradicting triples  $(x_i, x_j, x_k)$  and  $(x_i, x_k, x_j)$  cannot be present in  $\mathcal{S}$  at the same time. We will discuss how to deal with the general case below.

Our first kernel function is based on the following idea: We fix two objects  $x_a$  and  $x_b$ . In order to compute a similarity score between  $x_a$  and  $x_b$  we would like to rank all objects in  $\mathcal{D}$  with respect to their distance from  $x_a$  and also rank them with respect to their distance

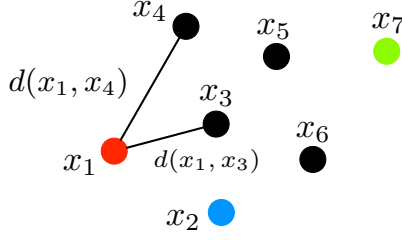


Figure 1: Illustration of the idea behind our first kernel function. In order to compute a similarity score between  $x_1$  (in red) and  $x_2$  (in blue) we would like to rank all objects with respect to their distance from  $x_1$  and also with respect to their distance from  $x_2$  and compute Kendall's  $\tau$  between the two rankings. In this example, the objects would rank as  $x_1 \prec x_3 \prec x_2 \prec x_4 \prec x_5 \prec x_6 \prec x_7$  and  $x_2 \prec x_3 \prec x_6 \prec x_1 \prec x_5 \prec x_4 \prec x_7$ , respectively. Kendall's  $\tau$  between these two rankings is  $1/3$ , and hence this would be the similarity score between  $x_1$  and  $x_2$ . For comparison, ranking the objects with respect to their distance from  $x_7$  (in green) yields  $x_7 \prec x_5 \prec x_6 \prec x_3 \prec x_4 \prec x_2 \prec x_1$ , and the similarity score between  $x_1$  and  $x_7$  would be  $-5/7$  and between  $x_2$  and  $x_7$  it would be  $-3/7$ .

from  $x_b$ , and take a similarity score between these two rankings as similarity score between  $x_a$  and  $x_b$ . One possibility to measure similarity between rankings is given by the famous Kendall tau correlation coefficient (Kendall, 1938), which is also known as Kendall's  $\tau$ : for two rankings of  $n$  items, Kendall's  $\tau$  between the two rankings is the fraction of concordant pairs of items minus the fraction of discordant pairs of items. Here, a pair of two items  $i_1$  and  $i_2$  is concordant if  $i_1 \prec i_2$  or  $i_1 \succ i_2$  holds true in both rankings, and discordant if they satisfy  $i_1 \prec i_2$  according to one and  $i_1 \succ i_2$  according to the other ranking. Formally, a ranking is represented by a permutation  $\sigma : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$  such that  $\sigma(i) \neq \sigma(j)$ ,  $i \neq j$ , and  $\sigma(i) = m$  means that item  $i$  is ranked at the  $m$ -th position. Given two rankings  $\sigma_1$  and  $\sigma_2$ , the fraction of concordant pairs equals

$$f_c(\sigma_1, \sigma_2) = \frac{1}{\binom{n}{2}} \cdot \sum_{i < j} [\mathbb{1}\{\sigma_1(i) < \sigma_1(j)\} \mathbb{1}\{\sigma_2(i) < \sigma_2(j)\} + \mathbb{1}\{\sigma_1(i) > \sigma_1(j)\} \mathbb{1}\{\sigma_2(i) > \sigma_2(j)\}], \quad (2)$$

the fraction of discordant pairs equals

$$f_d(\sigma_1, \sigma_2) = \frac{1}{\binom{n}{2}} \cdot \sum_{i < j} [\mathbb{1}\{\sigma_1(i) < \sigma_1(j)\} \mathbb{1}\{\sigma_2(i) > \sigma_2(j)\} + \mathbb{1}\{\sigma_1(i) > \sigma_1(j)\} \mathbb{1}\{\sigma_2(i) < \sigma_2(j)\}], \quad (3)$$

and Kendall's  $\tau$  between  $\sigma_1$  and  $\sigma_2$  is given by

$$\tau(\sigma_1, \sigma_2) = f_c(\sigma_1, \sigma_2) - f_d(\sigma_1, \sigma_2).$$

It has been established just recently that Kendall's  $\tau$  is actually a kernel function on the set of total rankings (Jiao and Vert, 2015). Consequently, by measuring similarity between the two rankings of objects (one with respect to their distance from  $x_a$  and one with respect to their distance from  $x_b$ ) we would not only compute a similarity score between  $x_a$  and  $x_b$ , but would even end up with a kernel function on  $\mathcal{D}$ . This idea is illustrated with an example in Figure 1.

In our situation, the problem is that in most cases  $\mathcal{S}$  will contain only a small fraction of all possible similarity triplets and also that some of the triplets in  $\mathcal{S}$  might be incorrect, so that there is no way of ranking all objects with respect to their distance from any fixed object based on the similarity triplets in  $\mathcal{S}$ . We therefore have to adapt the procedure. For doing so we consider the feature map that corresponds to the kernel function just described. Recall that a feature map corresponding to a kernel function  $k : \mathcal{D} \times \mathcal{D} \rightarrow \mathbb{R}$  is a mapping  $\Phi : \mathcal{D} \rightarrow \mathbb{R}^m$  for some  $m \in \mathbb{N}$  such that

$$k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle = \Phi(x_i)^T \cdot \Phi(x_j).$$

It is easy to see from (2) and (3) (also compare with Jiao and Vert (2015)) that the feature map corresponding to the described kernel function is given by  $\Phi_{k_\tau} : \mathcal{D} \rightarrow \mathbb{R}^{\binom{n}{2}}$  with

$$\Phi_{k_\tau}(x_a) = \frac{1}{\sqrt{\binom{n}{2}}} \cdot \left( \mathbb{1}\{d(x_a, x_i) < d(x_a, x_j)\} - \mathbb{1}\{d(x_a, x_i) > d(x_a, x_j)\} \right)_{1 \leq i < j \leq n}.$$

In our situation, where we are only given  $\mathcal{S}$  and cannot evaluate  $\Phi_{k_\tau}$  in most cases, we now have to replace  $\Phi_{k_\tau}$  by an approximation: up to a normalizing factor, we simply replace an entry in  $\Phi_{k_\tau}(x_a)$  by zero if we cannot evaluate it based on the similarity triplets in  $\mathcal{S}$ . More precisely, we consider the feature map  $\Phi_{k_1} : \mathcal{D} \rightarrow \mathbb{R}^{\binom{n}{2}}$  given by

$$\Phi_{k_1}(x_a) = \frac{1}{\sqrt{|\{(x_i, x_j, x_k) \in \mathcal{S} : x_i = x_a\}|}} \cdot \left( \mathbb{1}\{(x_a, x_i, x_j) \in \mathcal{S}\} - \mathbb{1}\{(x_a, x_j, x_i) \in \mathcal{S}\} \right)_{1 \leq i < j \leq n}, \quad (4)$$

and define our first proposed kernel function  $k_1 : \mathcal{D} \times \mathcal{D} \rightarrow \mathbb{R}$  by

$$k_1(x_i, x_j) = \langle \Phi_{k_1}(x_i), \Phi_{k_1}(x_j) \rangle = \Phi_{k_1}(x_i)^T \cdot \Phi_{k_1}(x_j). \quad (5)$$

Note that the scaling factor in the definition of  $\Phi_{k_1}$  in (4), ensuring that the feature embedding lies on the unit sphere, is crucial whenever the number of similarity triplets in which an object appears as anchor object is not approximately constant over the different objects. Also note that we have to assume that every object in  $\mathcal{D}$  appears at least once as an anchor object in a similarity triplet in  $\mathcal{S}$ . Otherwise,  $\Phi_{k_1}$  would not be well-defined since we would encounter a denominator equaling zero.

Our second kernel function is based on a similar idea, however, this time we do not consider  $x_a$  and  $x_b$  as anchor objects when measuring their similarity, but rather compare whether they rank similar with respect to their distances from the various other objects. Concretely, up to normalization, we would like to count the number of pairs of objects  $(x_i, x_j)$  for which the distance comparisons

$$d(x_i, x_a) \stackrel{?}{<} d(x_i, x_j)$$

and

$$d(x_i, x_b) \stackrel{?}{<} d(x_i, x_j)$$

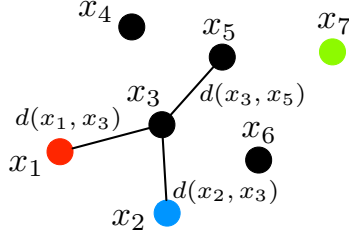


Figure 2: Illustration of the idea behind our second kernel function. In order to compute a similarity score between  $x_1$  (in red) and  $x_2$  (in blue) we would like to check for every pair of objects  $(x_i, x_j)$  whether the distance comparisons  $d(x_i, x_1) \stackrel{?}{<} d(x_i, x_2)$  and  $d(x_i, x_2) \stackrel{?}{<} d(x_i, x_1)$  yield the same result or not. In this example, we have 32 pairs for which they yield the same result (e.g.,  $(x_3, x_7)$  is one such a pair) and 17 pairs for which they do not (e.g.,  $(x_3, x_5)$ ). Hence, we could assign  $7^{-2} \cdot (32 - 17) = 15/49$  as similarity score between  $x_1$  and  $x_2$ . For comparison, the similarity score between  $x_1$  and  $x_7$  (in green) would be  $3/49$  and between  $x_2$  and  $x_7$  it would be  $1/49$ .

yield the same result and subtract the number of pairs for which these comparisons yield different results. This idea is illustrated in Figure 2. Adapted to our situation, where we are only given a non-exhaustive collection  $\mathcal{S}$  of similarity triplets, it corresponds to considering the feature map  $\Phi_{k_2} : \mathcal{D} \rightarrow \mathbb{R}^{n^2}$  given by

$$\Phi_{k_2}(x_a) = \frac{1}{\sqrt{|\{(x_i, x_j, x_k) \in \mathcal{S} : x_j = x_a \vee x_k = x_a\}|}} \cdot \left( \mathbb{1}\{(x_i, x_a, x_j) \in \mathcal{S}\} - \mathbb{1}\{(x_i, x_j, x_a) \in \mathcal{S}\} \right)_{1 \leq i \leq j \leq n} \quad (6)$$

and defining our second proposed kernel function  $k_2 : \mathcal{D} \times \mathcal{D} \rightarrow \mathbb{R}$  by

$$k_2(x_i, x_j) = \langle \Phi_{k_2}(x_i), \Phi_{k_2}(x_j) \rangle = \Phi_{k_2}(x_i)^T \cdot \Phi_{k_2}(x_j). \quad (7)$$

Again, the scaling factor in the definition of  $\Phi_{k_2}$  in (6) is crucial whenever there are objects occurring in more similarity triplets than others. Here we have to assume that for every object in  $\mathcal{D}$  there is at least one similarity triplet in which the object appears, but not as an anchor object.

### Contradicting similarity triplets

If  $\mathcal{S}$  contains contradicting triples  $(x_i, x_j, x_k)$  and  $(x_i, x_k, x_j)$  and there might be triples occurring repeatedly, one can either replace contradicting and multiple triples by just one triple representing a “majority decision” (if there is a tie, remove the triples completely) or one can alter the definition of  $\Phi_{k_1}$  or  $\Phi_{k_2}$  as follows: if  $\#\{(x_a, x_i, x_j) \in \mathcal{S}\}$  denotes the

number of how often the triple  $(x_a, x_i, x_j)$  occurs in  $\mathcal{S}$ , set

$$\Phi_{k_1}(x_a) = \frac{\tilde{\Phi}_{k_1}(x_a)}{\|\tilde{\Phi}_{k_1}(x_a)\|} \quad \text{where}$$

$$\tilde{\Phi}_{k_1}(x_a) = \left( \frac{\#\{(x_a, x_i, x_j) \in \mathcal{S}\} - \#\{(x_a, x_j, x_i) \in \mathcal{S}\}}{\#\{(x_a, x_i, x_j) \in \mathcal{S}\} + \#\{(x_a, x_j, x_i) \in \mathcal{S}\}} \right)_{1 \leq i < j \leq n}.$$

The definition of  $\Phi_{k_2}$  can be revised in an analogous way. In doing so, we incorporate a simple estimate of the likelihood of a triple being correct.

### Combining the two kernel functions

Clearly, one can combine  $k_1$  and  $k_2$  in order to obtain another kernel functions: for parameters  $\mu_1, \mu_2 > 0$  we define  $k_3^{\mu_1, \mu_2} : \mathcal{D} \times \mathcal{D} \rightarrow \mathbb{R}$  as

$$k_3^{\mu_1, \mu_2} = \mu_1 k_1 + \mu_2 k_2,$$

and there are further possibilities for building up new kernel functions from existing ones, for example  $k_1 \cdot k_2$  or  $\exp(k_i)$ ,  $i = 1, 2$  (Shawe-Taylor and Cristianini, 2004).

### Reducing diagonal dominance

If the number  $|\mathcal{S}|$  of given similarity triplets is small, our kernel functions suffer from a problem that is shared by many other kernel functions defined on complex data: the feature maps  $\Phi_{k_1}$  and  $\Phi_{k_2}$  map the objects in  $\mathcal{D}$  to sparse vectors, that is almost all of their entries are zero. As a consequence, two different feature vectors  $\Phi_{k_i}(x_a)$  and  $\Phi_{k_i}(x_b)$  appear to be almost orthogonal and the similarity score  $k_i(x_a, x_b)$  is much smaller than the self-similarity scores  $k_i(x_a, x_a)$  or  $k_i(x_b, x_b)$ . This phenomenon, usually referred to as diagonal dominance of the kernel function, has been observed to pose difficulties for the kernel methods the kernel function is applied to, and several ways have been proposed for dealing with it (Schölkopf et al., 2002; Greene and Cunningham, 2006). In all our experiments we deal with diagonal dominance in the following simple way: Let  $k$  denote a kernel function and  $K$  the kernel matrix on  $\mathcal{D}$ , that is  $K = (k(x_i, x_j))_{i,j=1}^n$ , which would be the input to a kernel method. Then we replace  $K$  by  $K - \lambda_{\min} I$  where  $I \in \mathbb{R}^{n \times n}$  denotes the identity matrix and  $\lambda_{\min}$  is the smallest eigenvalue of  $K$ . Note that  $\lambda_{\min}$  is non-negative and that it is the largest number that we can subtract from the diagonal of  $K$  such that the resulting matrix is still positive semi-definite and hence can be used by a kernel method.

### Meaningfulness of our kernel functions

Intuitively, our kernel functions measure similarity between  $x_a$  and  $x_b$  by quantifying to which extent  $x_a$  and  $x_b$  can be expected to be located in the same region of  $\mathcal{D}$ : Think of  $\mathcal{D}$  as being

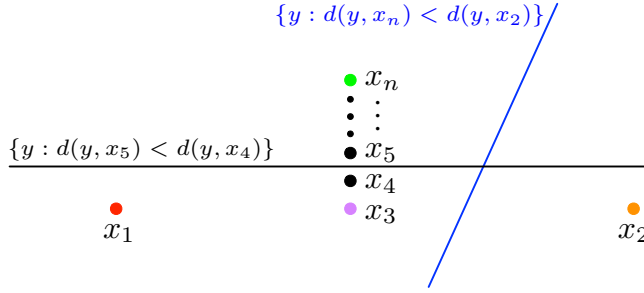


Figure 3: The kernel function  $k_1$  measures similarity between two objects essentially by counting in how many of the halfspaces that are obtained from distance comparisons the two objects reside at the same time. The outcome does not only depend on the distance between the two objects but also on their location within the data set: although  $x_1$  and  $x_2$  are located far apart from each other, the kernel function  $k_1$  considers them to be very similar. See the running text for details.

a subset of  $\mathbb{R}^m$  and  $d$  being the Euclidean metric. A similarity triplet  $d(x_a, x_i) < d(x_a, x_j)$  then tells us that  $x_a$  resides in the halfspace defined by the hyperplane that is perpendicular to the line segment connecting  $x_i$  and  $x_j$  and goes through the segment's midpoint. If there is also a similarity triplet  $d(x_b, x_i) < d(x_b, x_j)$ ,  $x_a$  and  $x_b$  thus are located in the same halfspace (assuming the correctness of the similarity triplets) and this is reflected by a higher value of  $k_1(x_a, x_b)$ . Similarly, a similarity triplet  $d(x_i, x_a) < d(x_i, x_j)$  tells us that  $x_a$  is located in a ball with radius  $d(x_i, x_j)$  centered at  $x_i$  and the value of  $k_2(x_a, x_b)$  is higher if there is a similarity triplet  $d(x_i, x_b) < d(x_i, x_j)$  telling us that  $x_b$  is located in this ball too and it is smaller if there is a similarity triplet  $d(x_i, x_j) < d(x_i, x_b)$  telling us that  $x_b$  is not located in this ball. Note that the similarity scores between  $x_a$  and  $x_b$  defined by our kernel functions do not only depend on the distance  $d(x_a, x_b)$  between  $x_a$  and  $x_b$  but also on how the points in  $\mathcal{D}$  are spread in the space and on the locations of  $x_a$  and  $x_b$  within  $\mathcal{D}$  since this affects how the various hyperplanes or balls are related to each other. Consider the example illustrated in Figure 3: Let  $d(x_3, x_n) = 1$  implying that  $d(x_i, x_{i+1}) = \Theta(1/n)$ ,  $3 \leq i < n$ , and  $d(x_1, x_2) > d(x_2, x_n) > d(x_1, x_n) > d(x_2, x_3) > d(x_1, x_3) > 1$  be arbitrarily large. Although  $x_1$  and  $x_2$  are located at the maximal distance to each other, they satisfy  $d(x_1, x_i) < d(x_1, x_j)$  and  $d(x_2, x_i) < d(x_2, x_j)$  for all  $3 \leq i < j \leq n$ , and hence both  $x_1$  and  $x_2$  are jointly located in all the halfspaces obtained from these distance comparisons. Note that these halfspaces can be arranged as a sequence of increasing subsets. It is easy to see that we end up with  $k_1(x_1, x_2) \rightarrow 1$ ,  $n \rightarrow \infty$ , assuming  $k_1$  is computed based on all possible similarity triplets, all of which are correct. On the other hand, the distance between  $x_3$  and  $x_n$  is much smaller, but there are many points in between them and the hyperplanes obtained from the distance comparisons with these points separate  $x_3$  and  $x_n$ . We end up with  $k_1(x_3, x_n) \rightarrow -1$ ,  $n \rightarrow \infty$ . Depending on the task at hand, this may be desirable or not.

Let us examine the meaningfulness of our kernel functions by calculating them on several visualizable data sets. Each data set consists of 400 points in  $\mathbb{R}^2$  and  $d$  equals the Euclidean metric. We computed the kernel functions  $k_1$ ,  $k_2$  and  $k_3^{1,1}$  based on 3176040 similarity triplets

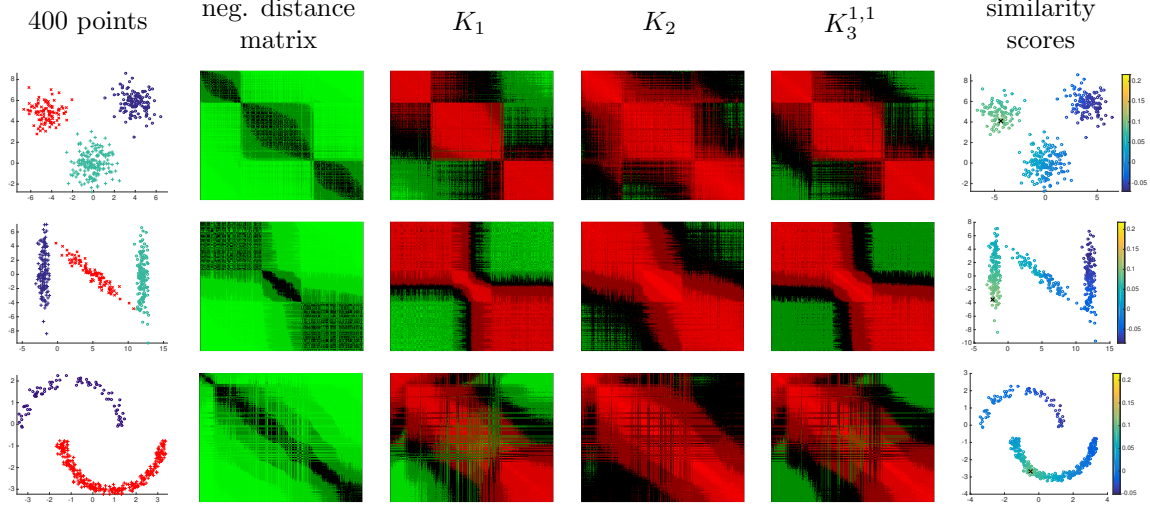


Figure 4: Kernel matrices for various data sets consisting of 400 points based on 10% of all similarity triplets. 1st plot: The data points. 2nd plot: The negated distance matrix. 3rd / 4th / 5th plot: The kernel matrix corresponding to  $k_1$  /  $k_2$  /  $k_3^{1,1}$ . 6th plot: Similarity scores (encoded by color) based on  $k_1$  between a fixed point (shown as a black cross) and the other points.

that were chosen uniformly at random without replacement from the set of all possible similarity triplets (the number of chosen triplets corresponds to 10% of all triplets). The results are shown in Figure 4. Each row corresponds to one data set. The first plot of a row shows the data set. The second plot shows the negated distance matrix on the data set. Next, we can see the kernel matrices. The last plot of a row shows the similarity scores based on  $k_1$  between one fixed point (shown as a black cross) and the other points in the data set. Clearly, the kernel matrices reflect the block structures of the negated distance matrices. As hoped for, the similarity scores are the smaller the larger the distances from the fixed points are. A situation like in the example of Figure 3 does not seem to occur.

## Computational complexity

A naive implementation of our kernel functions explicitly computes the feature vectors  $\Phi_{k_1}(x_i)$  or  $\Phi_{k_2}(x_i)$ ,  $i = 1, \dots, n$ , and subsequently calculates the kernel matrix  $K$  by means of (5) or (7). In doing so, we store the feature vectors in the feature matrix

$$\Phi_{k_1}(\mathcal{D}) = (\Phi_{k_1}(x_i))_{i=1}^n \in \mathbb{R}^{\binom{n}{2} \times n}$$

or

$$\Phi_{k_2}(\mathcal{D}) = (\Phi_{k_2}(x_i))_{i=1}^n \in \mathbb{R}^{n^2 \times n}.$$

Proceeding this way is straightforward and simple, requiring to go through  $\mathcal{S}$  only once, but comes with a computational cost of  $\mathcal{O}(|\mathcal{S}| + n^4)$  operations. Note that the number of different



distance comparisons of the form (1) is  $\mathcal{O}(n^3)$  and hence one might expect that  $|\mathcal{S}| \in \mathcal{O}(n^3)$  and  $\mathcal{O}(|\mathcal{S}| + n^4) = \mathcal{O}(n^4)$ . By performing (5) or (7) in terms of matrix multiplication  $\Phi_{k_1}(\mathcal{D})^T \cdot \Phi_{k_1}(\mathcal{D})$  or  $\Phi_{k_2}(\mathcal{D})^T \cdot \Phi_{k_2}(\mathcal{D})$  and applying Strassen’s algorithm (Higham, 1990) one can slightly reduce the number of operations to  $\mathcal{O}(|\mathcal{S}| + n^{3.81})$ , but still such a naive implementation is infeasible for any somewhat larger data set. However, this is also the case for ordinal embedding algorithms, which are the current state-of-the-art method for solving problems based on similarity triplets. All existing ordinal embedding algorithms iteratively solve an optimization problem. For none of these algorithms theoretical bounds for their complexity are available in the literature, but simulations in Kleindessner and von Luxburg (2016) show that their running time is tremendously high. We will also see in the experiments below that computing our kernel functions actually takes much less time than computing an ordinal embedding.

We believe that our kernel functions bear some potential regarding efficiency that should be examined in future work: On the one hand, if the number of given similarity triplets  $|\mathcal{S}|$  is small, then the feature matrix  $\Phi_{k_1}(\mathcal{D})$  or  $\Phi_{k_2}(\mathcal{D})$  is sparse with only  $\mathcal{O}(|\mathcal{S}|)$  non-zero entries and methods for sparse matrix multiplication decrease computational complexity (Yuster and Zwick, 2005). On the other hand, if  $\mathcal{S}$  is assumed to have some particular “structure”, that is one knows which distance comparisons were evaluated for the similarity triplets in  $\mathcal{S}$  and they are sorted in an appropriate way, one can do better than the naive implementation and compute the kernel matrix  $K$  by going through  $\mathcal{S}$  on the fly without explicitly computing the feature vectors  $\Phi_{k_1}(x_i)$  or  $\Phi_{k_2}(x_i)$ . Similarly, considering an active setting instead of the batch setting assumed in this paper (compare with the next section), by applying an appropriate query strategy one might be able to calculate the kernel matrix  $K$  without computing the feature vectors simultaneously to querying similarity triplets. However, a crucial question regarding all these suggestions is *how* small  $|\mathcal{S}|$  can be such that the kernel functions are still meaningful (also compare with the next section).

## Related work and further background

In the introduction we have defined similarity triplets as binary answers to distance comparisons

$$d(A, B) \stackrel{?}{<} d(A, C).$$

As such they are a special case of answers to the more general distance comparisons

$$d(A, B) \stackrel{?}{<} d(C, D), \quad A, B, C, D \in \mathcal{X}.$$

We refer to any collection of answers to these general comparisons as ordinal distance information. In recent years, a whole new branch of the machine learning literature has emerged that deals with ordinal distance information, both on the empirical as well as on the theoretical side. Among the work on ordinal distance information in general (see Kleindessner and von Luxburg (2016) for references), similarity triplets have been paid particular attention, mainly due to the fact that they can easily be gathered from humans: Given vector data

and similarity triplets for it, Schultz and Joachims (2003) learn a parameterized distance function that best explains the given triplets. Jamieson and Nowak (2011) deal with the important question of how many similarity triplets are required for uniquely determining an ordinal embedding of Euclidean data. Algorithms for constructing an ordinal embedding based on similarity triplets (but not on general ordinal distance information) are proposed in Tamuz et al. (2011) and van der Maaten and Weinberger (2012). Although the title of Tamuz et al. (2011) bears resemblance to the title of our paper, the aim of their paper is indeed rather different from ours: by kernel they mean a kernel function corresponding to a low-dimensional ordinal embedding as feature map. Heikinheimo and Ukkonen (2013) present a method for medoid estimation based on statements “Object  $A$  is the outlier within the triple of objects  $(A, B, C)$ ”, which correspond to the two similarity triplets  $d(B, C) < d(B, A)$  and  $d(C, B) < d(C, A)$ , and subsequently apply it to devise a crowdclustering algorithm. In Ukkonen et al. (2015), the authors use the same kind of statements for density estimation. Pretty useful for working with similarity triplets in practice might be the work by Wilber et al. (2014), who examine how to minimize time and costs when collecting similarity triplets via crowdsourcing. Producing a number of ordinal embeddings at the same time, each corresponding to a different dissimilarity function based on which a distance comparison (1) might have been evaluated, is studied in Amid and Ukkonen (2015). Kleindessner and von Luxburg (2016) propose algorithms for several problems based on statements “Object  $A$  is the most central object within the triple of objects  $(A, B, C)$ ”, which comprise the two similarity triplets  $d(B, A) < d(B, C)$  and  $d(C, A) < d(C, B)$ .

Despite the considerable number of papers, to the best of our knowledge we are the first to propose kernel functions that can be evaluated given only ordinal distance information about a data set (and that do not arise as a byproduct of an ordinal embedding) and thus make it possible to apply any kernel method to the data set. Let us make some more comments about ordinal distance information and similarity triplets.

### Some remarks on ordinal distance information and similarity triplets

Dealing with ordinal distance information comes with a critical drawback compared to the standard setting of cardinal distance information: while for a data set comprising  $n$  objects there are in total “only”  $\Theta(n^2)$  distances between objects, there are potentially  $\Theta(n^4)$  different distance comparisons involving four objects at a time and still  $\Theta(n^3)$  different comparisons involving three objects (corresponding to similarity triplets). Unless  $n$  is rather small, in practice it is thus prohibitive to collect answers to all possible comparisons. However, the hope is that not all distance comparisons are required and that relatively few of them already contain the bulk of usable information due to high redundancy. This gives rise to distinguishing between a batch setting where one is given an arbitrary collection of similarity triplets for a data set as we are assuming in this paper and an active setting in which one can query similarity triplets and aims at reducing their number by asking only for answers to the most informative distance comparisons (Jamieson and Nowak, 2011; Tamuz et al., 2011). Furthermore, the performance of any method based on similarity triplets has to be examined with respect to the number of similarity triplets it is provided as input. The performance also has to be examined with respect to the amount of incorrect similarity

triplets it can deal with: a method is the more useful the fewer similarity triplets it requires and the larger fraction of incorrect ones it allows for in order to produce a correct result. In the next section we experimentally examine our proposed kernel functions with respect to these two quantities.

## Experiments

We performed several experiments on both artificial and real data that demonstrate the meaningfulness and usefulness of our proposed kernel functions.

### Artificial data

We applied the kernel functions  $k_1$ ,  $k_2$  and  $k_3^{1,1}$  to various artificial data and different kernel methods. We compared our proposed approaches, that is applying one of our kernel functions to a kernel method, to straightforward ordinal embedding approaches. For constructing the ordinal embedding we used the SOE (soft ordinal embedding) algorithm (Terada and von Luxburg, 2014) and made use of its R implementation provided by the authors. In doing so, we set all parameters except the dimension of the space of the embedding to the provided default parameters. Collections of similarity triplets  $\mathcal{S}$ , provided as input to both our kernel functions and the SOE algorithm, were created as follows: Based on a dissimilarity function  $d$ , which we specify in each experiment, we created answers to all possible distance comparisons (1) (their number is  $|\mathcal{D}| \cdot (|\mathcal{D}| - 1) \cdot (|\mathcal{D}| - 2) / 2$ ). Answers were incorrect with some error probability  $0 \leq \text{errprob} \leq 1$  independently from each other. We then chose similarity triplets in  $\mathcal{S}$  from this set of answers uniformly at random without replacement. An answer being incorrect with error probability  $\text{errprob}$  means that a comparison (1) whose true answer is  $d(A, B) < d(A, C)$  yields the answer  $d(A, B) > d(A, C)$  with probability  $\text{errprob}$ .

### Clustering

We began with clustering a simple data set consisting of 300 points from a mixture of three Gaussians in  $\mathbb{R}^2$  similar to the one shown in the first row of Figure 4. The dissimilarity function  $d$  equaled the Euclidean metric. For assessing the quality of a clustering we computed its purity. Purity measures the accordance between an inferred clustering and a known ground truth partitioning of a data set  $\mathcal{D}$  as follows: if  $\mathcal{D}$  consists of  $L$  different classes  $C_1, \dots, C_L$  that we would like to recover and the clustering  $\mathcal{C}$  comprises  $K$  different clusters  $U_1, \dots, U_K$ , then the purity of  $\mathcal{C}$  is given by

$$\text{purity}(\mathcal{C}) = \text{purity}(\mathcal{C}; \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{k=1}^K \max_{l=1, \dots, L} |U_k \cap C_l|. \quad (8)$$

We always have  $K/|\mathcal{D}| \leq \text{purity}(\mathcal{C}) \leq 1$  and a high value indicates a good clustering. Figure 5 shows the purity of the clusterings obtained from kernel  $k$ -means (Dhillon et al., 2001) based

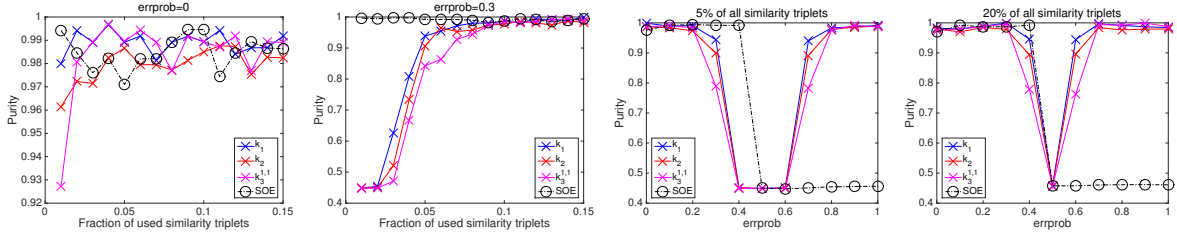


Figure 5: Clustering 300 points from a mixture of three Gaussians in  $\mathbb{R}^2$  with  $d$  being the Euclidean metric. Purity (8) (average over 100 runs) for kernel  $k$ -means based on  $k_1$  (in blue),  $k_2$  (in red) and  $k_3^{1,1}$  (in pink) and for ordinary  $k$ -means applied to the SOE embedding of the data points (in black). 1st / 2nd plot: Purity as a function of the fraction of the number of input similarity triplets compared to the number of all possible similarity triplets. 3rd / 4th plot: Purity as a function of *errprob*.

on our kernel functions and obtained from ordinary  $k$ -means applied to the SOE embedding. The results were averaged over 100 runs. We always provided the correct number of clusters, that is three, as input. The dimension of the space of the SOE embedding was correctly chosen as two. In the first and in the second plot we can study the purity of the various clusterings as a function of the fraction of the number of provided input similarity triplets compared to the number of all possible similarity triplets. If *errprob* = 0 (first plot), then all considered methods seem to perform similarly. Only when they are provided only 1% of all similarity triplets as input, the embedding approach seems to be slightly superior. The situation is different when *errprob* = 0.3 (second plot). In this case the embedding approach clearly outperforms our kernel functions, which can compete only when the fraction of input similarity triplets is 10% or larger. But note that the data set, since being Euclidean, is ideally suited for the embedding approach. In the third and in the fourth plot we can see the purity of the various clusterings as a function of *errprob* when either 5% (third plot) or 20% (fourth plot) of all possible similarity triplets were provided as input. As expected, for all methods the purity decreases as *errprob* increases from 0 to 0.5. Interestingly, the purity of kernel  $k$ -means based on our kernel functions then increases again. In hindsight, this is not surprising: if *errprob* = 1 and thus every similarity triplet is incorrect, we simply end up with the feature map  $-\phi_{k_1}$  or  $-\phi_{k_2}$ , when  $+\phi_{k_1}$  or  $+\phi_{k_2}$  are the feature maps based on only correct triplets, and hence with the same kernel function as for *errprob* = 0. When only 5% of all similarity triplets are provided as input, the embedding approach outperforms our kernel functions in the sense that it yields an almost perfect result even for *errprob* = 0.4 while for our kernel functions this is only the case as long as *errprob* < 0.3. For a larger number of input similarity triplets, namely 20% of all triplets, the difference between the performance of the embedding approach and our kernel functions becomes smaller. This is in agreement to the first and second plot.

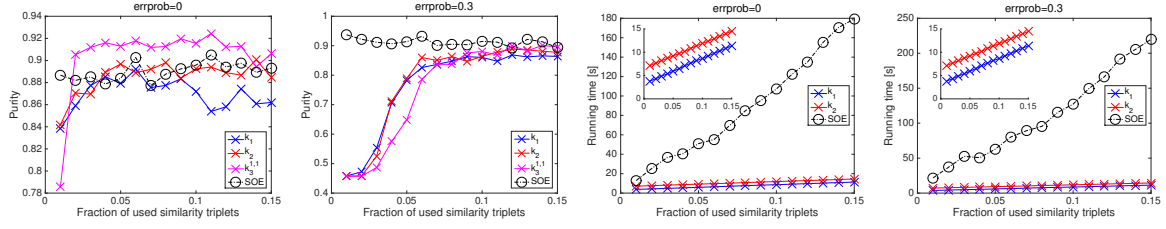


Figure 6: Clustering 300 points from a mixture of three Gaussians in  $\mathbb{R}^2$  with  $d$  being the shortest-path distance in the unweighted Euclidean minimum spanning tree on the points. 1st / 2nd plot: Purity (8) (average over 100 runs) as a function of the fraction of the number of input similarity triplets compared to the number of all possible similarity triplets for kernel  $k$ -means based on  $k_1$  (in blue),  $k_2$  (in red) and  $k_3^{1,1}$  (in pink) and for ordinary  $k$ -means applied to the SOE embedding of the data points (in black). 3rd / 4th plot: Running time in seconds (average over 100 runs) for computing  $k_1$  (in blue) and  $k_2$  (in red) and the SOE embedding (in black) as a function of the fraction of the number of input similarity triplets (all computations performed in R).

## Clustering non-Euclidean data

In this experiment we used the same set of points as in the previous one, that is 300 points from a mixture of three Gaussians in  $\mathbb{R}^2$ , but this time the dissimilarity function  $d$  did not equal the Euclidean metric. Instead, we chose  $d$  as the shortest-path distance in the Euclidean minimum spanning tree on the data points. For computing the shortest-path distance we considered the spanning tree as an unweighted graph. The first and the second plot in Figure 6 show the purity (average over 100 runs) of the clusterings obtained from kernel  $k$ -means based on  $k_1$ ,  $k_2$  and  $k_3^{1,1}$  and obtained from ordinary  $k$ -means applied to the SOE embedding as a function of the fraction of the number of input similarity triplets compared to the number of all possible similarity triplets. Again, we provided the correct number of clusters, that is three, as input and the dimension of the space of the SOE embedding was chosen as two. The results look similar to the ones of the previous experiment although both the embedding approach and our kernel functions perform worse than in the Euclidean case. In case of  $errprob = 0$  (first plot), all methods seem to perform similarly, with kernel  $k$ -means based on the kernel function  $k_3^{1,1}$  being slightly superior. In case of  $errprob = 0.3$  (second plot), our kernel functions require a significantly higher number of similarity triplets in order to produce a good clustering. Our kernel functions seem to be more sensitive to noise in the similarity triplets than the embedding approach — at least in these experiments.

## Running time

In order to make a fair comparison between the running times for computing our kernel functions and for computing an ordinal embedding we have implemented our kernel func-

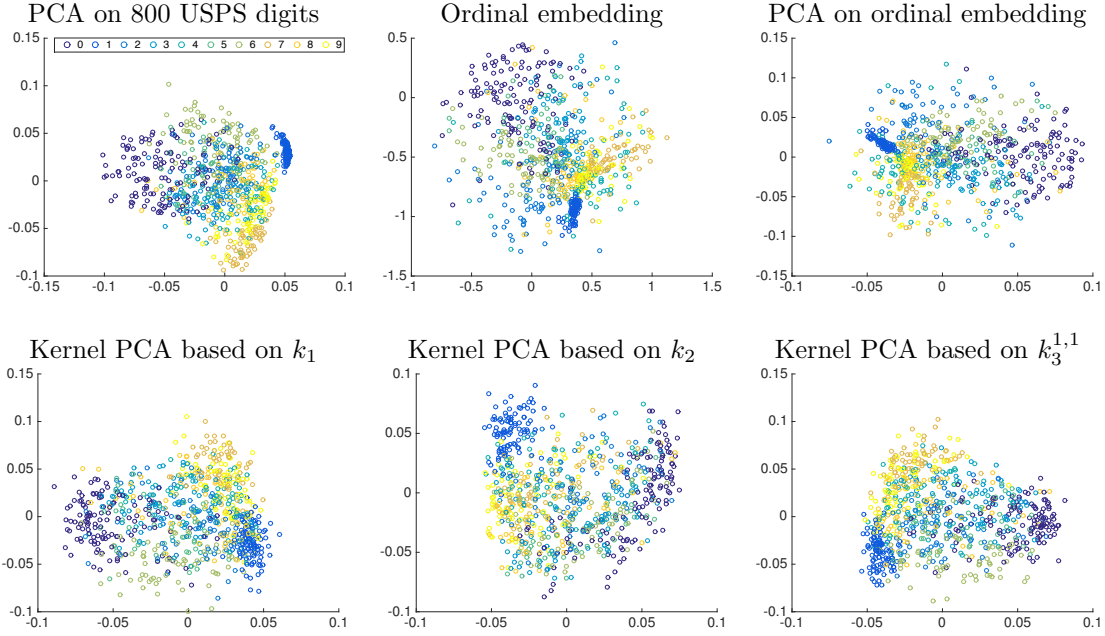


Figure 7: Various embeddings of 800 USPS digits with  $d$  equaling the Euclidean metric. The ordinal embedding and the kernel functions were computed based on 1% of all possible similarity triplets, which were incorrect with  $errprob = 0.3$

tions  $k_1$  and  $k_2$  in R. In the third and the fourth plot in Figure 6 we can see the running times for computing  $k_1$  or  $k_2$  using our R implementations in the course of the previous experiment. The plots also show the running time for computing the SOE embedding using the R implementation provided by Terada and von Luxburg (2014). The computations were performed on an iMac with 2.9 GHz Intel Core i5 and 8 GB 1600 MHz DDR3. We can see that computing our kernel functions is approximately ten times faster than computing the SOE embedding in case of  $errprob = 0$  (third plot) and the difference is even more severe in case of  $errprob = 0.3$  (fourth plot).

### Principal component analysis

We used our kernel functions  $k_1$ ,  $k_2$  and  $k_3^{1,1}$  to apply kernel PCA (Schölkopf et al., 1999) to a randomly chosen subset of 800 USPS digits. Similarity triplets were created based on  $d$  equaling the Euclidean metric. Figure 7 shows in the second row the projections of the data points onto the first two principal components when similarity triplets were incorrect with  $errprob = 0.3$  and we used 1% of all possible similarity triplets. For comparison, in the first row we can see the ordinary PCA embedding of the data points (using their coordinates), a two-dimensional ordinal embedding computed with the SOE algorithm from the same similarity triplets that we provided to our kernel functions and the ordinary PCA embedding of the points of this ordinal embedding. In our opinion, none of the shown embeddings looks superior. In all of them, the data points are somewhat clustered according to their labels,

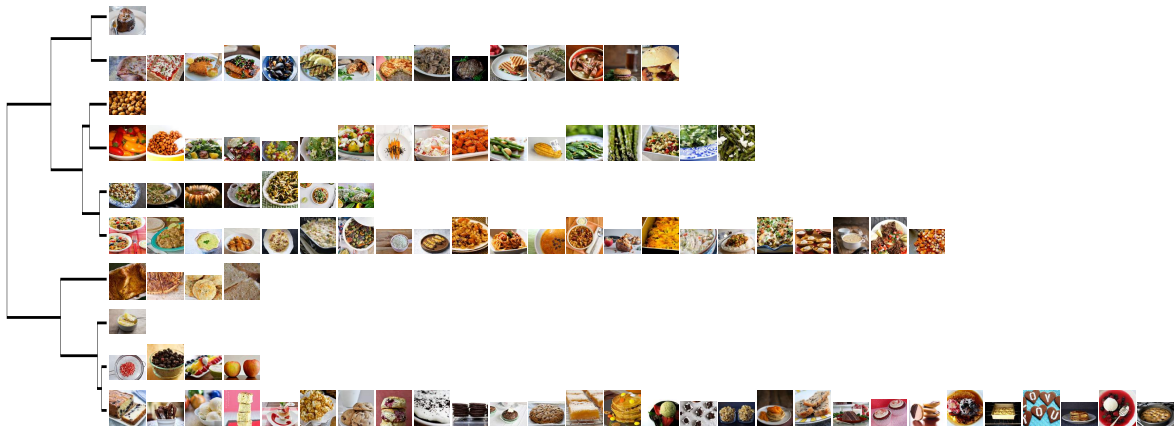


Figure 8: Clustering of the food data set. Part of the dendrogram obtained from complete-linkage clustering with the kernel function  $k_1$ .

but there is also significant overlap. All embeddings looked similar when we increased the number of input similarity triplets and also when we decreased the value of *errprob* (plots omitted). The only notable difference was that in case of  $errprob = 0$  the dark blue cluster in the kernel PCA embeddings was compressed in the same way as in the other embeddings.

## Real data

### Food data set

We applied the kernelized version of complete-linkage clustering based on our proposed kernel functions to the food data set introduced in Wilber et al. (2014). This data set consists of 100 images from a wide range of foods<sup>1</sup>. Encoding similarity triplets by ordered triples as in the second section, the authors have gathered a total of 190376 unique triples (which corresponds to a number of about 39% of the number of all possible similarity triplets) via crowd sourcing. However, there are 9349 contradicting pairs of triples. Figure 8 shows a part of the dendrogram that we obtained when working with the kernel function  $k_1$ . Each of the ten clusters depicted there contains pretty homogeneous images. For example, the last row only shows desserts whereas the fourth row only shows vegetables and salads. The same was true when working with the kernel functions  $k_2$  or  $k_3^{1,1}$  (figures omitted), and we consider the result to provide supporting evidence for the meaningfulness of our kernel functions.

### Car data set

In this experiment, we applied kernel PCA to the car data set introduced in Kleindessner and von Luxburg (2016). It consists of 60 images of cars, and the authors have collected state-

<sup>1</sup>According to the authors, the data set contains copyrighted material under the educational fair use exemption to the U.S. copyright law.

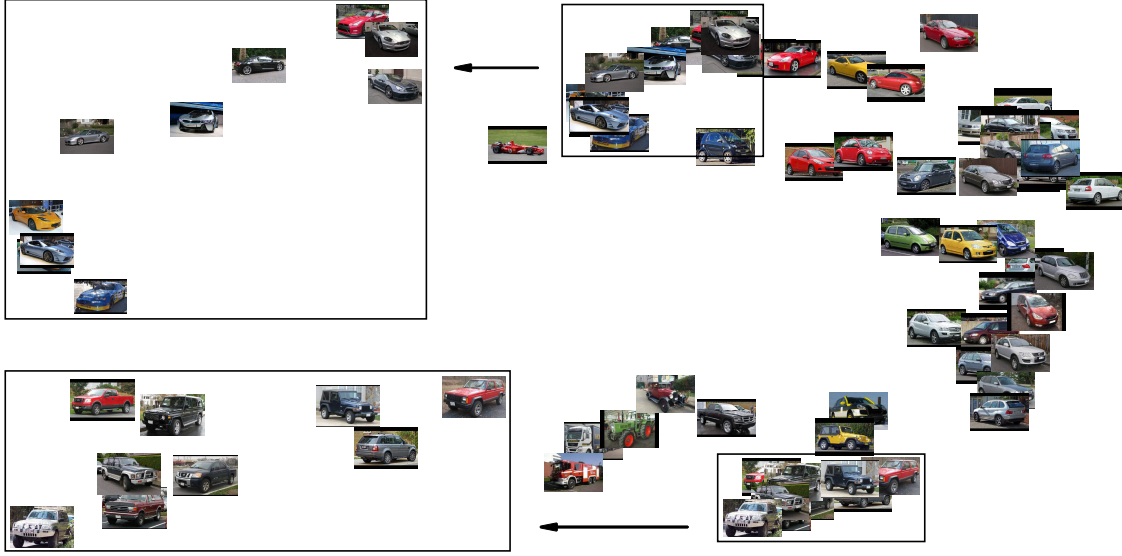


Figure 9: Kernel PCA on the car data set. Projection onto the first two principal components based on the kernel function  $k_2$ .

ments of the kind “Object  $A$  is the most central object within the triple of objects  $(A, B, C)$ ” for these images. One such a statement comprises two similarity triplets:  $d(B, A) < d(B, C)$  and  $d(C, A) < d(C, B)$ . Working with the statements in T\_ALL (one of four collections of statements provided by the authors) and encoding similarity triplets by ordered triples, we ended up with 13514 triples of which 12502 were unique (corresponding to a number of about 12% of the number of all possible similarity triplets). Again, there was a number of contradicting pairs of triples. The projection of the data set onto the first two principal components can be seen in Figure 9. For producing this figure we have used the kernel function  $k_2$ , but using  $k_1$  or  $k_3^{1,1}$  instead yielded similar ones (figures omitted). The result looks quite reasonable, with the cars obviously arranged in groups of sports cars (top left), ordinary cars (middle right) and off-road/sport utility vehicles (bottom left). Also within these groups there is some reasonable structure. For example, the race-like sports cars are located near to each other and close to the Formula One car and the red cars at the top are strikingly close. Again, we consider this experiment as supporting our claim that our proposed kernel functions are meaningful and useful.

## Discussion

In this paper we have proposed kernel functions that can be evaluated given only similarity triplets about a data set  $\mathcal{D}$ . These kernel functions can be applied to any kernel method in order to solve various machine learning problems. As opposed to existing methods, our kernel functions correspond to high-dimensional embeddings of  $\mathcal{D}$ . Their construction aims at quantifying the relative difference in the locations of two objects in  $\mathcal{D}$ . In a number of experiments we have demonstrated the meaningfulness of our kernel functions. Even with a



naive implementation they run much faster than current state-of-the-art methods.

## Acknowledgements

This work has been supported by the Institutional Strategy of the University of Tübingen (Deutsche Forschungsgemeinschaft, ZUK 63).

## References

- S. Agarwal, J. Wills, L. Cayton, G. Lanckriet, D. Kriegman, and S. Belongie. Generalized non-metric multidimensional scaling. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2007.
- E. Amid and A. Ukkonen. Multiview triplet embedding: Learning attributes in multiple maps. In *International Conference on Machine Learning (ICML)*, 2015.
- E. Arias-Castro. Some theory for ordinal embedding. arXiv:1501.02861 [math.ST], 2015.
- I. Dhillon, Y. Guan, and B. Kulis. Kernel k-means, spectral clustering and normalized cuts. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, 2001.
- D. Greene and P. Cunningham. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *International Conference on Machine Learning (ICML)*, 2006.
- H. Heikinheimo and A. Ukkonen. The crowd-median algorithm. In *Conference on Human Computation and Crowdsourcing (HCOMP)*, 2013.
- N. Higham. Exploiting fast matrix multiplication within the level 3 BLAS. *ACM Transactions on Mathematical Software*, 16(4):352–368, 1990.
- K. Jamieson and R. Nowak. Low-dimensional embedding using adaptively selected ordinal data. In *Allerton Conference on Communication, Control, and Computing*, 2011.
- Y. Jiao and J.-P. Vert. The Kendall and Mallows kernels for permutations. In *International Conference on Machine Learning (ICML)*, 2015.
- M. Kendall. A new measure of rank correlation. *Biometrika*, 30(1–2):81–93, 1938.
- M. Kleindessner and U. von Luxburg. Uniqueness of ordinal embedding. In *Conference on Learning Theory (COLT)*, 2014.
- M. Kleindessner and U. von Luxburg. Lens depth function and  $k$ -relative neighborhood graph: versatile tools for ordinal data analysis. arXiv:1602.07194 [stat.ML]. Data available on <http://www.wsi.uni-tuebingen.de/lehrstuehle/theory-of-machine-learning/people/matthaeus-kleindessner.html>, 2016.

- B. Schölkopf, A. Smola, and K.-R. Müller. Kernel principal component analysis. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods : Support Vector Learning*, pages 327–352. MIT Press, Cambridge, MA, 1999.
- B. Schölkopf, J. Weston, E. Eskin, C. Leslie, and W. Noble. A kernel approach for learning from almost orthogonal patterns. In *European Conference on Machine Learning (ECML)*, 2002.
- M. Schultz and T. Joachims. Learning a distance metric from relative comparisons. In *Neural Information Processing Systems (NIPS)*, 2003.
- J. Shawe-Taylor and A. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, 2004.
- O. Tamuz, C. Liu, S. Belongie, O. Shamir, and A. Kalai. Adaptively learning the crowd kernel. In *International Conference on Machine Learning (ICML)*, 2011.
- Y. Terada and U. von Luxburg. Local ordinal embedding. In *International Conference on Machine Learning (ICML)*, 2014. Code available on <https://cran.r-project.org/web/packages/loe>.
- A. Ukkonen, B. Derakhshan, and H. Heikinheimo. Crowdsourced nonparametric density estimation using relative distances. In *Conference on Human Computation and Crowdsourcing (HCOMP)*, 2015.
- L. van der Maaten and K. Weinberger. Stochastic triplet embedding. In *IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 2012.
- M. Wilber, I. Kwak, and S. Belongie. Cost-effective hits for relative similarity comparisons. In *Conference on Human Computation and Crowdsourcing (HCOMP)*, 2014. Data available on <http://vision.cornell.edu/se3/projects/cost-effective-hits/>.
- R. Yuster and U. Zwick. Fast sparse matrix multiplication. *ACM Transactions on Algorithms*, 1(1):2–13, 2005.